General Certificate of Education (International)
Advanced Level and Advanced Subsidiary Level

Syllabus

COMPUTING 9691

For examination in June and November 2010

Note for Exams Officers: Before making Final Entries, please check availability of the codes for the components and options in the E3 booklet (titled “Procedures for the Submission of Entries”) relevant to the exam session. Please note that component and option codes are subject to change.

COMPUTING 9691

GCE Advanced Subsidiary Level and GCE Advanced Level, for examination in 2010

CONTENTS

Exclusions

This syllabus must not be offered in the same session with any of the following syllabuses:

9754 Computing (Singapore)
9713 Applied Information and Communication Technology

INTRODUCTION

This syllabus is designed to give greater flexibility both to teachers and to candidates.

It is envisaged that students will utilise the skills and knowledge of computing in one of three ways. Firstly, to provide a general understanding and perspective of the use of computer technology and systems, which will inform their decisions and support their participation in an increasingly technologically dependent society. Secondly to provide the necessary skills and knowledge to seek employment in areas that utilise computing. Thirdly, students may continue to develop their knowledge and understanding of computing through entry to higher education, where this qualification will provide a useful foundation for further study of computing or more specialist aspects of computing.

Centres and candidates may choose:

- to take all Advanced Level components in the same examination session leading to the full A Level.
- to follow a **staged** assessment route to the Advanced Level by taking the Advanced Subsidiary (AS) qualification in an earlier examination session. Subject to satisfactory performance such candidates are then only required to take the final part of the assessment (referred to in this syllabus as A2) leading to the full A Level.
- to take the Advanced Subsidiary (AS) qualification only.

AIMS

The aims of a course based on this syllabus, whether leading to an AS or A Level qualification, should be to:

1. develop an understanding of the main principles of solving problems using computers;
2. develop an understanding of the range of applications of computers and the effects of their use;
3. develop an understanding of the organisation of computer systems including software, data, hardware, communications and people;
4. acquire the skills necessary to apply this understanding to developing computer-based solutions to problems.

In addition, an aim of a course leading to the full A Level qualification should be to:

5. develop an understanding of the main principles of systems analysis and design, methods of problem formulation and planning of solutions using computers, and systematic methods of implementation, testing and documentation.

ASSESSMENT OBJECTIVES

The assessment objectives are common to both the AS and A2 assessments.

A Knowledge with Understanding

Candidates should be able to:

1. describe and explain the impact of Computing in a range of applications and show an understanding of the characteristics of computer systems (hardware, software and communication) which allow effective solutions to be achieved;

2. describe and explain the need for and the use of various forms of data organisation and processing to support the information requirements of a particular application;

3. describe and explain the systematic development of high quality solutions to problems and the techniques appropriate for implementing such solutions;

4. comment critically on the social, legal, ethical and other consequences of the use of computers.

B Skills

Candidates should be able to:

1. analyse a problem and identify the parts which are appropriate for a computer-based solution;

2. select, justify and apply appropriate techniques and principles to develop data structures and algorithms for the solution of problems;

3. design, implement and document an effective solution using appropriate hardware, software and programming languages.

WEIGHTING OF ASSESSMENT OBJECTIVES

Paper	Percentage of Advanced Level		
	Assessment Objective A	Assessment Objective B	Total weighting
1	25 ± 2	10 ± 2	37.5
2	5 ± 2	10 ± 2	12.5
3	15 ± 2	15 ± 2	30
4	5 ± 2	15 ± 2	20

SCHEME OF ASSESSMENT

The Advanced Subsidiary GCE forms 50% of the assessment weighting of the full Advanced GCE. Advanced Subsidiary GCE is assessed at a standard between IGCSE and Advanced GCE and can be taken as a stand-alone course or as the first part of the full Advanced GCE course.

Assessment is by means of **two units** for Advanced Subsidiary GCE and **four units** for Advanced GCE.

Advanced Subsidiary GCE Candidates take papers 1 and 2.

Advanced GCE Candidates take papers 1, 2, 3 and 4.

Paper	Type of Paper	Duration	Marks	Weighting		
				AS	A2	A
1	Written paper	2 ½h	90	75	-	37.5
2	Practical Programming Project	-	50	25	-	12.5
3	Written paper	2h	90	-	60	30
4	Computing Project	-	60	-	40	20

Paper 1
This paper will consist of a variable number of compulsory questions of variable mark value. Candidates will answer on lined paper. This paper will be set according to the content of section 1 of the syllabus.

Paper 2
This paper is a practical programming project. Students should select a problem, the solution to which will enable them to demonstrate the skills required by the syllabus in Section 2.

The programming language used is at the discretion of the candidate and the Centre. CIE suggest that Visual Basic, Pascal, C++ and Java would all be suitable vehicles for the production of the software. However, this list is not exhaustive and if a candidate would like to use another language there should not be a problem. Centres can contact CIE Customer Services to ensure that another language is acceptable before beginning the work.

Candidates are free to select any problem for solution, but should discuss their chosen problem in detail with the staff at the Centre to ensure that the solution will demonstrate their abilities to the full.

Candidates may receive guidance in choosing their problem, but Centres should ensure that work from their candidates is sufficiently different to make them individual pieces of work.

This practical programming project should be completed during the first year of a two year course. It may be submitted, along with Paper 1, at the end of the first year in order to qualify for the award of AS in Computing, or may be saved and submitted at the end of the two years, in addition to Paper 1, 3 and the project to qualify for the award of A Level Computing. In this way, Centres wishing to enter candidates for Papers 1 and 3 can complete this coursework during the first year of the course. The projects will be marked by Centres and moderated by CIE.

Paper 3
This paper will consist of a variable number of compulsory questions of variable mark value. Candidates will answer on lined paper. This paper will be set according to the content of Section 3 of the syllabus, but will also assume knowledge learned in Section 1.

Paper 4
Further details of the project are to be found in Section 4 of the syllabus.

COURSEWORK

AS

Section 2 – Practical Programming Project. (50 marks)
This unit examines knowledge and understanding as well as skills. The programming project is intended to allow candidates to demonstrate their competence in the skills of program design, development, testing and documentation.
The criteria which should be followed when producing their solution are clearly set out in Section 2 of the syllabus.
The practical programming project will be marked at the Centre according to the criteria outlined in the section *Guidance on Marking the Practical Programming Project,* which can be found at the back of this syllabus. The marking criteria will not change from year to year. The marked projects will be externally moderated by CIE. If Centres are uncertain about the appropriateness of a problem they should seek advice from CIE.

A2

Section 4 – Computing Project. (60 marks)
This unit assesses candidates' ability to develop a computer-based solution to a real life problem requiring the skills of analysis, design, development, testing, implementation and evaluation. Candidates should formulate the task in negotiation with their teacher. If Centres are uncertain about the appropriateness of a problem they should seek advice from CIE.

Assessment and Moderation.

All coursework is marked by the teacher and internally standardised by the Centre. Coursework is then submitted to CIE by the specified date. The purpose of moderation is to ensure that the standard for the award of marks in coursework is the same for each Centre, and that each teacher has applied the same standards appropriately across the range of candidates within the Centre.

Minimum Coursework Requirements.

If a candidate submits no work for a coursework unit, then the candidate should be indicated as being absent from that unit on the coursework mark sheets submitted to CIE. If a candidate completes any work for the coursework unit, then the work should be assessed according to the criteria and marking instructions, and the appropriate mark awarded, which may be 0 (zero).

Authentication.

As with all coursework, the teacher must be able to verify that the work submitted for assessment is the candidate's own work. Sufficient work must be carried out under direct supervision to allow the teacher to authenticate the coursework marks with confidence.
CIE are happy to rely on the professionalism of teachers to ensure fairness with this work.

Differentiation.

In the question papers, differentiation is achieved by setting questions which are designed to assess candidates at their appropriate levels of ability and which are intended to allow all candidates to demonstrate what they know, understand and can do.
In coursework, candidates should choose their project problem so that the work enables them to display positive achievement and that will allow them to demonstrate their full range of abilities.

Please copy the Coursework Assessment Summary Form at the back of this syllabus document and submit with both the Practical Programming Project and the Computing Project. The Candidate Record card for both the Practical Programming Project and the Computing Project should be attached to each candidate's submission.

STRUCTURE OF THE SYLLABUS

This syllabus is set out in the form of teaching sections. Each teaching section is assessed by its associated paper. The Advanced Subsidiary syllabus consists of teaching sections 1 and 2 only, and the Advanced Level syllabus consists of all four teaching sections.

This section of the specifications gives the subject content for each section, as shown below.

Section	Associated Paper	Section Title
1	1	Computer Systems, Communications and Software
2	2	Practical Programming Project
3	3	Systems Software Mechanisms, Machine Architecture, Database Theory, Programming Paradigms and Integrated Information Systems
4	4	Computing Project

Each section is presented as a set of sub-sections, each with details of content and associated learning outcomes. An indication of recommended prior knowledge is given for each section, together with details of any links to other sections.

Section 1: Computer Systems, Communications and Software is the foundation for all subsequent sections. It provides candidates with an understanding of the core aspects of computer systems, which is developed and enhanced in subsequent sections.

Section 2: Practical Programming Project requires candidates to demonstrate their skills in a programming language by selecting a problem to solve. The solution to the problem should encompass as many of the criteria listed in Section 2 of the syllabus as the candidate is capable of using. Candidates and Centres should be aware that demonstration of the skills will be necessary, within the context of the problem solution, in order to earn marks in the assessment.
It is envisaged that this project will be a long term piece of work to be completed during year one of the course. It will be submitted for moderation in the same session that the candidate offers Paper 1. In this way, Centres that choose to do Papers 1 and 3 at the end of year 2 of a two year course, can do one piece of coursework in each year.
It is envisaged that most candidates will use one of the languages listed on page 3. Other languages are welcomed, though CIE Customer Services should be consulted before the project is started.

Section 3: Systems Software Mechanisms, Machine Architecture, Database Theory, Programming Paradigms and Integrated Information Systems provides candidates with further ability to extend skills, knowledge and understanding of computing concepts, gained in Section 1, to a range of applications in which computer systems are used.

Section 4: Computing Project requires candidates to identify a well-defined user-driven problem, involving a third-party user, and to generate a solution. As for Section 2, this is done using software tools chosen by the candidate and may include a programming language, an appropriate applications package or other software. It is envisaged that work on the Project will begin in parallel with work on Section 3.

SUBJECT CONTENT

SECTION 1: COMPUTER SYSTEMS, COMMUNICATIONS AND SOFTWARE

This section provides candidates with an understanding of the following core aspects of computer systems:

components of a computer system and modes of use;
system software;
programming tools and techniques;
data: their representation, structure and management;
hardware;
data transmission and networking.

The systems development life cycle is studied with reference to particular applications. Therefore, candidates are expected to look at a range of different types of application areas. Although candidates are not expected to have specific knowledge of every one, candidates should be able to make use of relevant examples for the purpose of illustration. This section also provides candidates with understanding of the following aspects of computer systems:

systems development life cycle;
choosing applications software for application areas;
handling of data in information systems;
characteristics of information systems;
implications of computer use.

1.1 Components of a Computer System and Modes of Use

Content

1.1.1 Types of hardware
1.1.2 Types of software

Learning outcomes

Candidates should be able to:

(a) define the terms hardware, software, input device, storage device and output device;
(b) describe the purpose of input devices, storage devices and output devices;
(c) define the different types of software: operating system, user interface, translator, utilities, programming languages and generic/common applications software;
(d) describe the hardware used to enable computers to communicate.

1.2 System Software

Content

1.2.1 Operating systems
1.2.2 User interfaces
1.2.3 Utility software

Learning outcomes

Candidates should be able to:

(a) describe the purpose of operating systems;
(b) describe the characteristics of different types of operating systems and their uses: batch, real-time, single-user, multi-user and network systems;
(c) describe different types of user interface: forms, menus, GUI, natural language and command line, suggesting the characteristics of user interfaces which make them appropriate for use by different types of user;
(d) describe the purpose of the following types of utility software: disk formatting, file handling (deleting, copying, moving, sorting), hardware drivers, file compression and virus checkers.

1.3 Programming Tools and Techniques

Content

1.3.1 Problem-solving techniques
1.3.2 Features of procedural programming languages
1.3.3 Basic translation process
1.3.4 Program testing
1.3.5 Program maintenance

Learning outcomes

Candidates should be able to:

(a) solve problems in a structured way, using logic and reason;
(b) describe techniques for writing software, including the splitting up of a problem into small sections and the use of appropriate techniques showing step-wise refinement/top-down and bottom-up design;
(c) explain that translators are needed to convert source code to object code (detailed knowledge of the types of translator and the translation process is **not** required);
(d) describe and give examples of types of programming error (syntax, logic and arithmetic);
(e) design a test plan using different testing strategies such as white box testing, black box testing, alpha and beta-testing;
(f) select suitable test data for a given problem;
(g) describe and use appropriately the tools, techniques and methods available for identifying programming errors (translator diagnostics, debugging tools, desk checking, bottom up programming, test strategies);
(h) demonstrate an understanding of the need for the use of comments, meaningful data names, indentation and modularity, in order to facilitate the ongoing maintenance of programs;
(i) dry run an algorithm given in a pseudo code format;
(j) produce an algorithm, in whatever form, to solve a simple problem.

1.4 Data: Its Representation, Structure and Management

Content

1.4.1 Data types
1.4.2 Data structures
1.4.3 Data management

Learning outcomes

Candidates should be able to:

(a) explain the use of codes to represent a character set (e.g. ASCII and EBCDIC);
(b) explain the use of different data types: integer, Boolean, date/time, currency and character;
(c) express integers in binary form;
(d) express numbers in binary form;
(e) define and use arrays (single and multi-dimensional) for solving simple problems, including initialising arrays, reading data into arrays and performing a simple serial search on an array;
(f) define and use linked lists (single pointer) for solving simple problems, including initialising linked lists and performing a simple serial search on a linked list (detailed algorithms are not expected);
(g) describe the LIFO and FIFO features of the data structures, stacks and queues;
(h) explain how data is stored in files in the form of fixed length records comprising items in fields;
(i) design a record format;
(j) estimate the size of a file;
(k) define and explain the difference between serial, sequential, indexed sequential and random access to data, using examples and stating their comparative advantages and disadvantages;

(l) describe how serial, sequential and random organisation of files may be implemented using indexes and hashing algorithms as appropriate (detailed algorithms and procedures will not be required);

(m) select appropriate data types/data structures for a given problem and explain the advantages and disadvantages of alternative choices;

(n) explain the procedures involved in backing up data and archiving, including the difference between data that is backed up and data that is archived.

1.5 Hardware

Content

1.5.1 Processor components
1.5.2 Primary and secondary storage
1.5.3 Peripheral devices

Learning outcomes

Candidates should be able to:

(a) describe the function and purpose of the control unit, memory unit and arithmetic logic unit (ALU) as individual parts of a processor;

(b) explain the difference between types of primary memory and their uses (RAM, ROM);

(c) describe the basic features, advantages, disadvantages and use of secondary storage media, both magnetic and optical;

(d) describe use of buffers and interrupts in the transfer of data between secondary storage and primary memory;

(e) describe a range of common peripheral devices in terms of their features, advantages, disadvantages and uses;

(f) relate the choice of peripheral device to a given application.

1.6 Data Transmission and Networking

Content

1.6.1 Data transmission
1.6.2 Circuit switching and packet switching
1.6.3 Protocols
1.6.4 Networking

Learning outcomes

Candidates should be able to:

(a) describe the characteristics of a local area network (LAN) and a wide area network (WAN);

(b) show an understanding of the hardware and software needed for a local area network (LAN) and for accessing a wide area network (WAN);

(c) describe basic network topologies (bus, star and ring) explaining the benefits and drawbacks of each topology;

(d) describe the different types of data transmission: serial and parallel; and simplex, half duplex and duplex modes;

(e) explain the relationship between bit rates and the use of data content;

(f) recognise that errors can occur in data transmission, and explain the use of parity checks and check sums in detecting and correcting these errors;

(g) explain the difference between packet switching and circuit switching;

(h) define the term protocol;

(i) describe the need for communication between devices, and between computers, and explain the need for protocols to establish communication links (candidates will **not** be expected to have detailed knowledge of specific protocols);

(j) discuss the advantages and disadvantages of networking.

1.7 Systems Development Life Cycle

Content

1.7.1 Identification of problem
1.7.2 Feasibility study
1.7.3 Information collection
1.7.4 Analysis of a problem, based upon information collected, including producing a requirements specification
1.7.5 Design of system to fit requirements
1.7.6 Development and testing of system
1.7.7 Implementation of system
1.7.8 Maintenance of system
1.7.9 Obsolescence

Learning outcomes

Candidates should, with reference to particular applications, be able to:

(a) understand the system life cycle as an iterative process;
(b) explain the importance of defining a problem accurately;
(c) describe the function and purpose of a feasibility study;
(d) explain the importance of determining the information requirements of a system and describe different methods of fact finding, including questionnaires, observation, and structured interviews, highlighting the advantages and disadvantages of each method;
(e) describe what is involved when analysing the requirements of a system, explaining the nature of the requirements specification and its content, identify inefficiencies/problems, user requirements and hardware and software requirements;
(f) design the data structures, inputs, outputs and processing using diagrammatic representations where appropriate;
(g) explain the importance of evaluating the system against initial specifications;
(h) explain the content and importance of documentation in the system life cycle, including the requirements specification, design specification, program specifications, technical and user manuals;
(i) explain the importance of system testing and implementation planning;
(j) explain the purpose of maintaining the system, and explain the need for system review and reassessment, understanding that software has a limited life span.

1.8 Choosing Applications Software for Application Areas

Content

1.8.1 Custom written software versus off-the-shelf software packages
1.8.2 Application areas
1.8.3 Applications software

Learning outcomes

Candidates should, within context, be able to:

(a) distinguish between custom-written software and off-the-shelf software packages, and discuss the advantages and disadvantages of each in given situations;
(b) identify the features of common applications found in business, commercial and industrial applications: e.g. stock control, order processing, payroll, process control, point of sale systems, marketing, computer aided design (CAD), computer aided manufacture (CAM);
(c) identify suitable common generic applications software for particular application areas e.g. word processing, spreadsheets, desktop publishers (DTP), presentation software, drawing packages;
(d) identify application areas for which generic applications software is not appropriate;
(e) describe the purpose and impact of different types of generic applications software, for example word processing, spreadsheets, desktop publishers (DTP), presentation software, drawing packages;
(f) explain how a common software package can be used to solve a given, simple, problem.

1.9 Handling of Data in Information Systems

Content

1.9.1 Data capture, preparation and data input
1.9.2 Validation and verification of data
1.9.3 Outputs from a system

Learning outcomes

Candidates should, within a context, be able to:

(a) describe manual and automatic methods of gathering and inputting data into a system, including form design, keyboard entry, voice recognition, barcodes, optical mark recognition (OMR), magnetic stripe cards, optical character recognition (OCR), data logging, touch screens;

(b) describe image capture by use of a scanner, video capture card and digital camera/camcorder;

(c) explain the techniques of validation and verification, and describe validation tests which can be carried out on data;

(d) describe possible output formats such as graphs, reports, interactive presentations, sound, video, images and animations stating the advantages and disadvantages of each format;

(e) discuss the need for a variety of output formats according to the target audience. Knowledge of timeliness, relevance etc. of intended output is required.

1.10 Designing the User Interface

Content

1.10.1 Interface design
1.10.2 Criteria for selecting appropriate hardware

Learning outcomes

Candidates should be able to:

(a) discuss the importance of good interface design;

(b) select appropriate peripheral hardware, including special purpose input devices, for a given application, and justify the choice;

(c) discuss human computer interaction (HCI) design issues such as the use of colour, layout, and content;

(d) distinguish between different styles of interface, including forms, menus, command-line input, natural language, speech, direct manipulation, and their relevance to application design;

(e) identify the required characteristics of a user interface with respect to information, type of user, physical location and current technology;

(f) understand the potential problem of speed mismatch between user, peripheral and processor.

1.11 Characteristics of Information Systems

Content

1.11.1 Passive versus interactive systems
1.11.2 Characteristics and uses of management information systems
1.11.3 Batch processing and rapid response applications
1.11.4 Knowledge based systems

Learning outcomes

Candidates should, within a context, be able to:

(a) give examples of, and describe the differing characteristics of, passive information systems and interactive information systems;

(b) describe the characteristics and uses of management information systems (MIS);

(c) identify a range of applications requiring batch processing and applications in which a rapid response is required;

(d) describe the use of knowledge based (expert) systems in fault diagnosis, geological surveys and medical diagnosis.

1.12 Implications of Computer Use

Content

1.12.1 Economic implications
1.12.2 Social implications
1.12.3 Legal implications
1.12.4 Ethical implications

Learning outcomes

Candidates should be able to:

(a) discuss changing trends in computer use and their economic, social, legal and ethical effects on society;

(b) explain changes to society brought about by the introduction and use of computer systems (e.g. changing leisure patterns and work expectations);

(c) discuss steps which can be taken to protect confidentiality of data held on computer systems;

(d) understand the need for data protection legislation;

(e) discuss the social and ethical implications of access to information whose value is controversial;

(f) discuss health and safety implications (for example, repetitive strain injury (RSI)) of increased computer use, including measures to ensure a healthy and safe working environment for employees.

SECTION 2: PRACTICAL PROGRAMMING PROJECT

This section is designed to allow candidates to develop the following skills:

program design;
program development;
testing;
implementation.

This section covers basic knowledge and understanding, as well as skills. It is expected that candidates will have studied the requisite theory in order to carry out the project successfully.

The Practical Programming Project is an individual piece of well-documented work involving a problem that can be solved using a computing system. The emphasis is on the solution of problems in a structured way using logic and reason to split a problem into sections that can be programmed using a procedural or object-oriented programming language.

Candidates are free to choose problems/tasks identified by themselves or their teacher. The choice of problem/task must allow the candidate to demonstrate the following programming skills in one program:

arrays and/or records
different data types
selection
iteration
procedures
functions
searching techniques
files

Candidates may solve the same problem or use the same initial scenario for a project but **the solution must be developed on an individual basis**, no collaborative work is allowed.

Teachers are expected to give educational guidance during the design process but the work submitted must be the candidate's own. Only the code designed and written by the candidate should be marked by the teacher.

The teacher marks the projects using the marking criteria in the *Guidance on Marking Practical Programming Projects* section of this syllabus, after which moderation takes place according to CIE procedures.

Candidates should not submit magnetic or optical media as part of their supporting evidence.

2.1 Problem/Task Identification — 2 marks

Candidates should be able to describe a problem/task that can be solved by writing a program.

2.1.1 Problem/Task description

Learning outcomes

Candidates should be able to:

(a) describe a problem in terms of inputs, processes and outputs.

2.2 Program Design — 6 marks

Content

Candidates should be able to specify and document a design. The design specification may include the method of solving a problem, for example:

2.2.1 Hardware requirements
2.2.2 Input design
2.2.3 Output design
2.2.4 Data structures
2.2.5 Processes

Learning outcomes

Candidates should be able to:

(a) specify the required hardware for a problem/task;
(b) design and document screen layouts;
(c) design and document report layouts, screen displays and/or other forms of output (for example, audio output);
(d) design and document the data structures necessary to model a problem/task;
(e) design and document a process model.

2.3 Program Development — 28 marks

Content

2.3.1 Interpreting a design solution
2.3.2 Developing a programmed solution

Learning outcomes

Candidates should be able to:

(a) interpret a given process model;
(b) specify any variables and data structures needed in the solution of a problem;
(c) develop a solution using a programming language;
(d) develop inputs/outputs using the features of the programming language;
(e) make use of the commenting feature of the programming language, meaningful variable names, indentation and modularity.

2.4 Testing — 8 marks

Content

2.4.1 Test strategy
2.4.2 Test data
2.4.3 Testing a programmed solution

Learning outcomes

Candidates should be able to:

(a) identify, develop and document a test strategy for a given problem;
(b) select suitable test data for a given problem;
(c) test a software solution, providing documented evidence that the solution works.

2.5 Implementation — 6 marks

Content

2.5.1 Installation instructions
2.5.2 Technical Documentation

Learning outcomes

Candidates should be able to:

(a) prepare basic installation instructions;
(b) prepare basic technical documentation for the software solution.

SECTION 3: SYSTEMS SOFTWARE MECHANISMS, MACHINE ARCHITECTURE, DATABASE THEORY, PROGRAMMING PARADIGMS AND INTEGRATED INFORMATION SYSTEMS

The content includes:

the functions of operating systems;
the functions and purposes of translators;
computer architectures and the fetch-execute cycle;
data representation, data structures and data manipulation;
programming paradigms;
databases;
use of systems and data;
systems development, implementation, management and applications;
simulation and real-time processing;
common network environments, connectivity and security issues.

Recommended Prior Knowledge

Candidates should have studied Section 1.

3.1 The Functions of Operating Systems

Content

3.1.1 Features of operating systems
3.1.2 Scheduling
3.1.3 Interrupt handling
3.1.4 Job queues and priorities
3.1.5 Memory management
3.1.6 Spooling
3.1.7 Modern personal computer operating systems

Learning outcomes

Candidates should be able to:

(a) describe the main features of operating systems: memory management, scheduling algorithms and distributed systems;
(b) explain how interrupts are used to obtain processor time and how processing of interrupted jobs may later be resumed (typical sources of interrupts should be identified and any algorithms and data structures should be described);
(c) define and explain the purpose of scheduling, job queues, priorities and how they are used to manage job throughput;
(d) explain how memory is managed in a typical modern computer system (virtual memory, paging and segmentation should be described);
(e) describe spooling, explaining why it is used;
(f) describe the main components of a typical desktop PC operating system;
(g) describe the main components of a network operating system including transparency, directory services, security and network printing.

3.2 The Functions and Purposes of Translators

Content

3.2.1 Types of translators
3.2.2 Lexical analysis
3.2.3 Syntax analysis
3.2.4 Code generation
3.2.5 Linkers and loaders

Learning outcomes

Candidates should be able to:

(a) describe the difference between interpretation and compilation;
(b) describe what happens during lexical analysis;

(c) describe what happens during syntax analysis, explaining how errors are handled;
(d) explain the code generation phase;
(e) explain the purpose of linkers and loaders.

3.3 Computer Architectures and the Fetch-Execute Cycle

Content

3.3.1 Von Neumann architecture
3.3.2 Registers: purpose and use
3.3.3 Fetch-execute cycle
3.3.4 Parallel processors

Learning outcomes

Candidates should be able to:

(a) describe basic Von Neumann architecture, identifying the need for and the uses of special registers in the functioning of a processor;
(b) describe in simple terms the fetch/decode/execute/reset cycle and the effects of the stages of the cycle on specific registers;
(c) discuss parallel processor systems, their uses, and their advantages and disadvantages.

3.4 Data Representation, Data Structures and Data Manipulation

Content

3.4.1 Number systems
3.4.2 Floating point binary
3.4.3 Normalisation of floating point binary numbers
3.4.4 Implementation of data structures, including linked lists, stacks, queues and trees
3.4.5 Searching and sorting

Learning outcomes

Candidates should be able to:

(a) express numbers in binary coded decimal (BCD), octal and hexadecimal;
(b) describe and use two's complement and sign and magnitude to represent positive and negative integers;
(c) perform integer binary arithmetic: addition and subtraction;
(d) demonstrate an understanding of floating point representation of a real binary number;
(e) normalise a floating point binary number;
(f) discuss the trade-off between accuracy and range when representing numbers in floating point form;
(g) explain the difference between static and dynamic implementation of data structures, highlighting the advantages and disadvantages of each;
(h) describe algorithms for the insertion and deletion of data items stored in linked-list, stack and queue structures;
(i) describe the insertion of data items into a sorted binary tree structure;
(j) explain the difference between binary searching and serial searching, highlighting the advantages and disadvantages of each;
(k) explain the difference between insertion sort and merge sort;
(l) describe algorithms for implementing insertion sort and merge sort methods;
(m) describe the use of a binary tree to sort data.

3.5 Programming Paradigms

Content

3.5.1 Types of languages and typical applications
3.5.2 Features of different types of language
3.5.3 Methods for defining syntax

Learning outcomes

Candidates should be able to:

(a) describe, with the aid of examples, the characteristics of a variety of programming paradigms (low level, object-orientated, declarative and procedural);
(b) explain, with examples, the terms object-oriented, declarative and procedural as applied to high-level languages;
(c) explain how functions, procedures and their related variables may be used to develop a program in a structured way, using stepwise refinement;
(d) describe the use of parameters, local and global variables as standard programming techniques;
(e) explain how a stack is used to handle procedure calling and parameter passing;
(f) discuss the concepts and, using examples, show an understanding of data encapsulation, classes and derived classes, and inheritance when referring to object-oriented languages;
(g) discuss the concepts and, using examples, show an understanding of backtracking, instantiation and satisfying goals when referring to declarative languages;
(h) explain the concepts of direct, indirect and indexed addressing of memory when referring to low level languages;
(i) using examples, describe the nature and purpose of 3rd and 4th generation languages;
(j) explain the need for, and be able to apply, BNF (Backus-Naur form) and syntax diagrams.

Note: Candidates will **not** be expected to use any particular form to present algorithms, but should be able to write procedural algorithms in some form.
Candidates will **not** be expected to write or interpret the meaning of simple segments of low level language code.
A detailed knowledge of the syntax of programming languages is **not** required.

3.6 Databases

Content

3.6.1 Database design
3.6.2 Normalisation and data modelling
3.6.3 Methods and tools for analysing and implementing database design
3.6.4 Control of access to relational database elements

Learning outcomes

Candidates should be able to:

(a) describe flat files, network, hierarchical and relational databases;
(b) design a simple relational database to the third normal form (3NF), defining tables and views of data;
(c) draw entity-relationship (E-R) diagrams to represent diagrammatically the data model;
(d) explain the advantages that using a relational database gives over flat files;
(e) define and explain the purpose of primary, secondary and foreign keys;
(f) explain the importance of varying the access allowed to database elements at different times and for different categories of user;
(g) describe the structure of a database management system (DBMS), including the function and purpose of the data dictionary, data description language (DDL) and data manipulation language (DML).

3.7 Use of Systems and Data

Content

3.7.1 The commercial value of data
3.7.2 The importance of standards
3.7.3 Communications and electronic commerce
3.7.4 Training
3.7.5 Effects of introducing systems

Learning outcomes

Candidates should be able to:

(a) identify data which has commercial value, explaining why such data has this value, and discuss contemporary trends in the compilation and use of valuable databases;

(b) explain the advantages of standardisation and describe some areas of standardisation such as file formats, ISDN, OSI model and its use together with communications protocols;

(c) describe ways in which computers aid communication, including voice mail, e-mail, digital telephone system facilities, e-commerce over the internet, tele/videoconferencing and electronic data interchange;

(d) identify situations in which the transmission of data, for example over the Internet, has created or could create new opportunities for businesses and individuals, in particular explaining how e-commerce works;

(e) identify and describe training and re-training requirements for a given situation;

(f) describe the substantial short-term and long-term changes, in patterns of work and in quality of output, which occur as a result of introducing computing systems.

3.8 Systems Development, Implementation, Management and Applications

Content

3.8.1 Methodologies and software tools for system development
3.8.2 Application types and technical requirements
3.8.3 Choice of implementation approaches: direct, parallel, phased
3.8.4 Systems management and monitoring

Learning outcomes

Candidates should, within the context of a scenario, be able to:

(a) identify commonly used techniques and software tools (e.g. Gantt charts, critical path analysis software and Entity Relationship diagrams) used in developing computer systems;

(b) describe how use of methodologies/techniques and software tools for developing computer systems aid the systems analyst/designer and programmer in terms of the documentation, step-by-step logical progression through tasks and cross-checking mechanisms;

(c) discuss the technical requirements of a system necessary to implement a range of different computer applications, including hardware, operating systems, communications, interface software, and other utility software;

(d) explain the need to provide appropriate response times for different applications and its implications for hardware, software and data structures;

(e) select, plan and justify appropriate implementation approaches for a range of different applications such as: parallel, phased, pilot, direct;

(f) discuss the implications of managing, monitoring and maintenance of systems, including the need for up-to-date documentation, software audit, quality control and management and hardware updates.

3.9 Simulation and Real-time Processing

Content

3.9.1 Applications of real-time computing
3.9.2 The feedback loop; input and output; sensors and actuators
3.9.3 The use of robots
3.9.4 Uses of simulation
3.9.5 Variation of parameters and conditions; time steps
3.9.6 Processing requirements
3.9.7 Advantages and limitations of simulations

Learning outcomes

Candidates should be able to:

(a) describe real-time applications;
(b) explain the use of sensors and actuators for visible, tactile, audible and other physical signals;
(c) demonstrate an understanding of the use of robots in a variety of situations such as the manufacturing process or hazardous environments;
(d) explain the reasons for simulation, such as to change time-scales and/or save costs and/or avoid danger;
(e) explain the large processing requirements of some systems and hence recognise the need for parallel architectures;
(f) discuss the advantages of simulation in testing the feasibility of a design.

3.10 Common Network Environments, Connectivity and Security Issues

Content

3.10.1 Data transmission
3.10.2 Network components
3.10.3 Use of networks to support hyperlinking systems such as the World Wide Web (WWW)
3.10.4 Common network environments
3.10.5 Issues of confidentiality
3.10.6 Encryption and authentication techniques

Learning outcomes

Candidates should be able to:

(a) describe methods used to organise LANs and WANs, and typical rates of data transmission associated with different topologies and methods;
(b) demonstrate awareness of different media for transmitting data and their carrying capabilities;
(c) explain the different purposes of network components, including switches, routers, bridges and modems;
(d) discuss common network environments, such as intranets, the Internet and other open networks, their facilities, structure and ability to exchange information using appropriate software and techniques;
(e) describe the purpose of hypertext linking, identifying the means by which it can be achieved such as hotwords/links, buttons and hypertext mark up language (HTML);
(f) describe the basic features of mark up languages;
(g) describe the facilities provided by electronic mail systems: responding, filing, copying, attaching, sending on and multiple recipients;
(h) discuss the problem of maintaining confidentiality of data on an open network and how to address this problem;
(i) explain the need for encryption, authorisation and authentication techniques (candidates will not be expected to know any specific method in detail).

SECTION 4: COMPUTING PROJECT

The project is a substantial piece of work requiring analysis and design over an extended period of time, which is organised, evaluated and presented in a report.

Candidates choose, in conjunction with their teacher, a well-defined user-driven problem which enables them to demonstrate their skills in Analysis, Design, Software Development, Testing, Implementation, Documentation and Evaluation.

Projects should be chosen to demonstrate the integrative aspects of the work and should avoid needless repetition of the demonstration of a given skill. Each candidate must submit a report on their piece of work, supported by evidence of software development and testing.

The teacher marks the projects using the marking criteria in the *Guidance on Marking Projects* section of this syllabus, after which moderation takes place according to CIE procedures.

4.1 Definition, Investigation and Analysis — 11 marks

Explanation of the problem to be solved, the user's requirements and how they were obtained. There should be a clear statement of requirements, agreed with the prospective user.

Content

4.1.1 Define a problem
4.1.2 Investigate the current system
4.1.3 Record findings
4.1.4 Analyse findings
4.1.5 Identify problems/inefficiencies with current system
4.1.6 Specify requirements: user, hardware, software

Learning outcomes

Candidates should be able to:

(a) define the nature of the problem to be solved;
(b) identify methods by which to investigate the problem: including questionnaires, observation and structured interviews;
(c) record information/data and gather sample documents currently used;
(d) identify the current processes and current data structures;
(e) analyse the data and processes: candidates will be expected to use appropriate techniques such as structure diagrams/data flow diagrams/system flowcharts to illustrate their analysis;
(f) specify inefficiencies and problems apparent from discussions with the user and the analysis work that has been carried out;
(g) derive the user and information requirements of the system;
(h) specify the required hardware and give reasons for their choice;
(i) specify the required software and give reasons for their choice;
(j) develop and document a clear requirement specification.

4.2 Design — 11 marks

Detailed system design including data structures, input-output format and processes involved. There should be a clear design specification.

Content

4.2.1 Output design
4.2.2 Input design
4.2.3 Data structures/model
4.2.4 Process model

Learning outcomes

Candidates should be able to:

(a) design and document report layouts, screen displays and/or other forms of output, drawing up detailed models of the proposed interface;
(b) design and document data capture forms and/or screen layouts;
(c) design and document using appropriate techniques (for example, normalisation/E-R models) the data structures necessary to solve the inefficiencies/problems indicated in the requirements specification;
(d) design and document an algorithm/pseudocode/top-down diagram or other form of process model.

4.3 Software Development, Testing and Implementation 18 marks

A software solution and comprehensive test plan is developed from the design, which should show that the system works with valid, invalid and extreme data (or, if it does not, under which circumstances it fails). The test plan should be clearly cross-referenced to provide evidence that the system has been tested during development and implementation. User testing should be in evidence.

Content

4.3.1 Software development
4.3.2 Test strategy/plan
4.3.3 Test data
4.3.4 Testing a software solution and planning for its implementation
4.3.5 User testing

Learning outcomes

Candidates should be able to:

(a) implement the proposed process model using a software package and/or programming language;
(b) develop the data structures of the design using the appropriate features of a software package and/or programming language;
(c) develop inputs/outputs using the features of a software package and/or programming language;
(d) identify, develop and document a test strategy for the software solution;
(e) select suitable test data to carry out the test strategy;
(f) test the software solution, illustrating how the software solution evolves;
(g) produce detailed output from the testing, cross referencing as appropriate to the test plan;
(h) test the software solution with the user, providing documented evidence that the solution works, and devise a strategy for its implementation.

4.4 Documentation 12 marks

Content

4.4.1 Technical documentation
4.4.2 User guides

The **Technical Manual** should include an explanation of the structure of the design and the solution. All the necessary information about the system that would allow someone else to maintain and develop it should be included, for example, back up procedures/cycles, annotated code/modules, data structures used and how they may be modified etc.

The **User Manual** should include step by step instructions for operating all aspects of the system, including a means of dealing with any errors that may occur. As well as a guide, User Documentation should include appropriate “Help” and messages within the software solution, or be present in the form of a hypertext document.

Learning outcomes

Candidates should be able to:

(a) develop a detailed technical manual;
(b) develop a detailed user manual.

4.5 Evaluation 8 marks

Discussion of the degree of success in meeting the original objectives as specified in the requirements specification, ease of the use of the package, acceptability to the users (including where possible a letter of acceptance from the user and reference to user testing results) and desirable extensions.

Content

4.5.1 Evaluate results against the requirement specification
4.5.2 Evaluate user testing
4.5.3 Identify the good and bad points of the final system, including any limitations and necessary extensions to the system

Learning outcomes

Candidates should be able to:

(a) evaluate the final system against the criteria described in the requirements specification;
(b) evaluate the users' responses to testing the system;
(c) identify the good and bad points of the final system highlighting any limitations and necessary extensions to the system, indicating how the extensions could be carried out.

GUIDANCE ON MARKING THE PRACTICAL PROGRAMMING PROJECT (9691/02)

Practical Programming Projects are assessed as follows:

Problem/Task identification	**[2 marks]**
Program Design	**[6 marks]**
Program Development	**[28 marks]**
Testing	**[8 marks]**
Implementation	**[6 marks]**

(a) Problem/Task Identification **[Total: 2 marks]**

A candidate should not expect the Examiners to be familiar with the problem/task that has been chosen. There should be a brief description of the problem/task and a clear statement of the form of data input should be given together with the required output.

1 Outline of the problem to be solved.
2 Description of the problem to be solved including the data input and the desired output.

(b) Program Design **[Total: 6 marks]**

A detailed program design (including diagrams as appropriate) should be produced. Proposed record, file and data structures should be described. Design of input formats (with examples of screen layouts) and output formats should be included here. A detailed description of processes should also be included. The hardware requirements must be stated.

1–2 Some vague discussion of what the program will do with a brief diagrammatic representation.
3–4 There is an outline of a design specification, including mock ups of inputs and outputs, process model described (including a diagram: structure diagram, data flow diagram or system flowchart). However there is a lack of completeness with omissions from the process model, inputs and outputs. Data structures have been identified but there may be inadequate detail. Or there may be some errors or logical inconsistencies, for example validation specified may be inadequate or field lengths incorrect.
5–6 A detailed and complete design specification, which is logically correct. There are also detailed written descriptions of any processes/modules and a clear, complete definition of any data structures.

(c) Program Development **[Total: 28 marks]**

(i) Implementing the program **[6 marks]**

There is evidence that the program produces the desired results. The finished program should relate clearly to the design work.

1–2 Program listings are provided in the form of printouts. The developed solution does not fulfil the design specification. A teacher may award up to 2 marks if they have been shown the system working satisfactorily and there is no hard evidence in the project report.
3–4 Program listings are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. The developed solution has logical flaws and is only slightly related to the design.

5–6 Program listings are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is a full set of printouts showing input and output as well as data structures. The program is clearly related to the design. All hardcopy is fully annotated and cross-referenced.

(ii) Using Good Programming Style [6 marks]

Program listings should be easily readable. There should be a 'header' identifying the program that contains the program name, author, school or college, programming language used, revision number, date and purpose. The program should be self-documenting. All data declarations should have explanatory comments; identifiers should have meaningful variable names; programs, functions and procedures should be clearly named, well separated and fully commented; suitable indentation should be used to set out the programming constructs used.

Program listings must contain all the code written by the candidate. If any library routines or automatically generated code is included this must be clearly identified and not taken into account for assessment purposes.

1–2 Program listings are not easily readable and have few comments or comments are handwritten on the listing.

3–4 The program listing shows some attention to good style but not all elements are included.

5–6 The program listing is easily readable and shows considerable attention to good style.

(iii) Programming Skills [16 marks]

Candidates must demonstrate their use of the following programming skills.

arrays and/or records
different data types
selection
iteration
procedures
functions
searching techniques
files

For each of the above skills:

1 mark for a valid use

1 mark for correct annotation within the code

(d) Testing [Total: 8 marks]

It is the responsibility of the candidates to produce evidence of their development work and to produce a test plan for the system. It is vital to produce test cases and to show that they work. To do this it is necessary, not only to have test data, but to know what the expected results are with that data.
An attempt should be made to show that all parts of the program have been tested, including those sections dealing with unexpected or invalid data as well as extreme cases. Showing that many other cases of test data are likely to work – by including the outputs that they produce – is another important feature. Evidence of testing is essential. Comments by teachers and others are of value, but the test plan must be supported by evidence in the report of a properly designed testing process. The examiner must be left in no doubt the program actually works. This evidence may be in the form of hardcopy output (possibly including screen dumps), photographs or VHS video.

1–2 A collection of hardcopy test run outputs with no test plan, or a test plan with no hardcopy evidence may also be present. A teacher may award up to 2 marks if they have been shown the program working satisfactorily and there is no hard evidence in the project report.

3–4 There is little evidence of testing with a badly developed test plan with clear omissions. There is no description of the relationship between the structure of the development work and the testing in evidence.

5–7 The developed solution partially fulfils the design specification. There should be at least eight different test runs together with a test plan and hardcopy evidence. However, the test plan has omissions in it and/or not all cases have been tested.

8 A comprehensive test plan, with evidence of each test run is present in the report, together with the expected output. The test plan should cover all aspects of the programming designed to cover the topics in **c (iii)** and demonstrate their effective use within the boundaries of the solution.

(e) Implementation [Total: 6 marks]

(i) Technical Documentation [4 marks]

Much of the documentation will have been produced as a by-product of design and development work and also as part of writing up the report to date. The following should be included: record, file and data structures used; data dictionary; data flow (or navigation paths); annotated program listings; detailed flowcharts; details of the algorithms and formulae used. These should be fully annotated since this is important for subsequent development of the system. The specifications of the hardware and software on which the system was developed should be included.
Since the contents in the technical documentation will differ from one project to another, professional judgement as to what would be necessary for another analyst to maintain and develop the program has to be made.

1–2 Some items are present with some annotation attempted.

3 One or two omissions, but the rest is present and annotation is used sensibly.

4 No major omissions, with all parts fully annotated.

(ii) Installation Instructions [2 marks]

Clear guidance, as friendly as possible, should be given to the user on how to install the program ready for use.

1 Sensible instructions on how to install the program for use.

2 Comprehensive, well illustrated instructions on how to install the program for use.

COMPUTING

Advanced GCE
Unit 9691/02: Practical Programming Project

Candidate Record Card

Please read the instructions printed overleaf before completing this form. One of these cover sheets, suitably completed, should be attached to the assessed work of **each** candidate in the moderation sample.

Examination session	**June/November***	*please delete as necessary	**Year**	**2**	**0**	**1**	

Centre name	

Centre number					

Candidate name		**Candidate number**				

Assessment Criterion	**Mark**
Problem/Task Identification (max 2)	
Program Design (max 6)	
Program Development (max 28)	
Testing (max 8)	
Implementation (max 6)	
Total (max 50)	

Authentication by the teacher

I declare that, to the best of my knowledge, the work submitted is that of the candidate concerned. I have attached details of any assistance given beyond that which is acceptable under the scheme of assessment.

Signature _____ Date _____

UNIVERSITY *of* CAMBRIDGE
International Examinations

INSTRUCTIONS FOR COMPLETION OF THIS FORM

1 One form should be used for each candidate.

2 Please ensure that the appropriate boxes at the top of the form are completed.

3 Enter the mark awarded for each Assessment Criterion in the appropriate box.

4 Add together the marks for the Assessment Criteria to give a total out of 50. Enter this total in the relevant box.

5 Sign and date the form.

COMPUTING 9691/02
Coursework Assessment Summary Form
A/AS Level

Centre Number						Centre Name		

Candidate Number	Candidate Name	Teaching Group/Set	Total Mark (max 50)	Internally Moderated Mark (max 50)

Name of teacher completing this form		Signature		Date						
Name of Internal moderator										

UNIVERSITY of CAMBRIDGE
International Examinations

GUIDANCE ON SELECTING THE COMPUTING PROJECT (9691/04)

The selection of the problem for which a computerised system is to be designed and implemented is extremely important. It should be chosen by the candidate in consultation with the teacher, and should always involve a user, ideally a 'third-party' user.

It is important to stress that the candidate should endeavour to produce a system which is non-trivial and which will solve a given problem sensibly within the constraints of resources available to the candidate.

Since the computing project seeks to assess the systems analysis section of the specification in a practical manner, candidates should not produce a system from their own limited knowledge of the requirements of the system. The 'third-party' user has to be someone who is willing to be involved in the project:

- in the analysis of the problem, where the user's requirements are obtained. This may take the form of a recorded interview with the candidate;
- at the software development, testing and implementation stages, where the user is involved in 'prototyping';
- at the evaluation stage, where the user is involved in checking that the system is completed as specified and, leading on from this, is then willing to write a letter of acceptance of the system, including any criticisms of it.

In this way, candidates can be encouraged to look beyond school, or college, life into the businesses and companies in the community of the surrounding area. The emphasis is on analysing an existing system, and producing a computer-based solution to fit the needs of the user.

At the end of the project, candidates should submit a concisely written and well laid out report, which should be word-processed.

The solution may be implemented using one or more of: a programming language, pre-written modules or toolkits, applications software and programmable packages. Brief descriptions of any programming languages or software packages used, together with reasons for their selection, should be included in the report.

Where the solution has involved programming the candidate should:

- Annotate listings;
- Explain each section of the program with appropriate algorithm descriptions, which should be language independent;
- Define variables by name, type and function where appropriate;
- Define clearly and identify the purpose of subroutines and procedures.

Where the solution has been produced with a software package that has not involved programming, the candidate should:

- Explain each section of the solution with appropriate algorithm descriptions;
- Define the purpose and inter-relationship of modules within the system;
- Clearly annotate the results produced.

The projects should contain the title, a contents list, a description of and a justification of the investigation, analysis, design and the methods used. Also an evaluation and a bibliography.

Pages should be clearly annotated with a footer containing the candidate name and the page number.

Appropriate evidence of development, testing and implementation, such as screen dumps of photographs of screen layouts and printouts, paper based user documentation and a letter from the third-party user to say that the system has been developed satisfactorily, must support the report.

Candidates should not submit magnetic or optical media as supporting evidence.

The computing project may involve programming or the tailoring of generic software packages and may also involve the choosing and installing of hardware. It is not intended that any method of solution is better than another, merely that the solution must be one that suits the problem that is being solved.

GUIDANCE ON MARKING THE COMPUTING PROJECT (9691/04)

Computing projects are assessed as follows:

Definition, Investigation and Analysis	**[11 marks]**
Design	**[11 marks]**
Software Development, Testing and Implementation	**[18 marks]**
Documentation	**[12 marks]**
Evaluation	**[8 marks]**

(a) Definition, Investigation and Analysis **[Total: 11 marks]**

(i) Definition – nature of the problem **[3 marks]**

A candidate should not expect the Examiners to be familiar with the theory and practice in the area of the chosen system. There should be a brief description of the organisation (for example, firm or business) involved and the current methods used in the chosen areas that may form the basis of the project. A clear statement of the origins and form of data should be given. At this stage the exact scope of the project may not be known and it may lead to the arranging of an interview with the user.

1 Description of the organisation.

2 Description of the organisation and the methods currently used in the area of the chosen project.

3 Full description of the organisation and methods currently in use in the area of the chosen project, with a description of the origin of the data to be used and some indication of the form that data takes.

(ii) Investigation and Analysis **[8 marks]**

This section is the 'systems analysis'. The candidate should describe how the user requirements were ascertained (possibly by long discussions with the users: question and answer sessions should be recorded and outcomes agreed). A clear requirements specification should be defined. Alternative outline solutions should be discussed and evaluated against one another.

1–2 Some elements have been discussed but little or no user involvement.

3–4 Some evidence that an attempt has been made to interview the user and some recording of it has been made. An attempt has been made to develop a requirement specification based on the information collected.

5–6 Good user involvement and recording of the interview(s). Most of the necessary items have been covered including a detailed discussion of alternative approaches. A requirements specification based on the information collected is present but with some omissions.

7–8 Excellent user involvement with detailed recording of the user's requirements. Alternative approaches have been discussed in depth. The report demonstrates a thorough analysis of the system to be computerised. A detailed requirements specification based on the information collected has been produced.

(b) Design **[Total: 11 marks]**

(i) Nature of the solution **[7 marks]**

A detailed systems design (including diagrams as appropriate) should be produced and agreed with the users. Proposed record, file and data structures should be described and design limitations should be included. Design of data capture forms, input formats (with examples of screen layouts) and output formats should be included here where relevant. A detailed summary of the aims and objectives should also be included. These items are the design specifications, which should be agreed with the user.

1–2 Some vague discussion of what the system will do with a brief diagrammatic representation of the new system.

3–4 The major objectives of the new system have been adequately summarised, *but omissions have been made*. There is a brief outline of a design specification, including mock ups of inputs and outputs, process model described (including a diagram: structure diagram, data flow diagram or system flowchart). However there is a lack of completeness with omissions from the process model, inputs and outputs. Data structures have been identified but there may be inadequate detail.

5–6 A clear set of objectives have been defined and a full design specification is included but there may be some errors or logical inconsistencies, for example validation specified may be inadequate or field lengths incorrect. There is clear evidence that a response to the design has been obtained from the end-user, and any comments have been acted upon.

7 A clear set of objectives with a detailed and complete design specification, which is logically correct. There are also detailed written descriptions of any processes/modules and a clear, complete definition of any data structures. The specification is sufficient for someone to pick up and develop an end result using the software and hardware specified in the requirements specification.

(ii) Intended benefits [2 marks]

There should be some discussion of the relative merits of the intended system and of the previous mode of operation. This may include any degree of generality beyond the original scope of the system.

1 One valid benefit of the new system has been identified and explained.

2 The benefits of the new system have been comprehensively described.

(iii) Limits of the scope of the solution [2 marks]

This may include volume (sizing limitations) and limitations of the facilities used. For full marks there must be some estimate of the size of the files required for the implemented system.

1 A discussion of what the system limitations are.

2 A detailed description of the system limitations has been given, including the estimate of the size of the files required for the implemented system.

(c) Software Development, Testing and Implementation [Total: 18 marks]

(i) Development and Testing [9 marks]

A technical description of how the solution relates to the design specification produced and agreed with the user should be included. It is the responsibility of the candidates to produce evidence of their development work and for producing a test plan for the system. It is vital to produce test cases and to show that they work. To do this, it is necessary not only to have test data, but to know what the expected results are with that data.
An attempt should be made to show that all parts of the system have been tested, including those sections dealing with unexpected or invalid data as well as extreme cases. Showing that many other cases of test data are likely to work – by including the outputs that they produce – is another important feature. Evidence of testing is essential. Comments by teachers and others are of value, but the test plan must be supported by evidence in the report of a properly designed testing process. The examiner must be left in no doubt the system actually works in the target environment. This evidence may be in the form of hardcopy output (possibly including screen dumps), photographs or VHS video.

1–2 Program listings or evidence of tailoring of a software package is provided in the form of printouts but with no annotation or relationship to a test plan or test run. The developed solution does not fulfil the design specification. A collection of hardcopy test run outputs with no test plan, or a test plan with no hardcopy evidence may also be present. A teacher may award up to 2 marks if they have been shown the system working satisfactorily and there is no hard evidence in the project report.

3–4 Program listings or evidence of tailored software packages are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is some annotation evident to illustrate how the package was tailored for a particular purpose or to indicate the purpose of sections of code in a program listing. The developed solution has logical flaws and does not fulfil the design specification. There is little evidence of testing with a badly developed test plan with clear omissions. There is no description of the relationship between the structure of the development work and the testing in evidence.

5–7 Program listings or evidence of tailored software packages are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is some annotation evident to illustrate how the package was tailored for a particular purpose or to indicate the purpose of sections of code in a program listing. The developed solution partially fulfils the design specification. There should be at least eight different test runs together with a test plan and hardcopy evidence. However, the test plan has omissions in it and/or not all cases have been tested.

8–9 Program listings or evidence of tailored software packages are provided in the form of printouts. Data structures are illustrated as part of the listings where appropriate, detailing their purpose. There is a full set of printouts showing input and output as well as data structures. All hardcopy is fully annotated and cross-referenced. A comprehensive test plan, with evidence of each test run is present in the report, together with the expected output. The test plan should cover as many different paths through the system as is feasible, including valid, invalid and extreme cases. Marks may be lost for lack of evidence of a particular test run or lack of expected results.

(ii) Implementation [6 marks]

It is recognised that the user organisation (preferably 'third-party') may not fully implement the system, although this is the ultimate aim. However, to score any marks in this section there must be some evidence that the person for whom the system was written has seen the system in operation. This can be done in a number of ways: such as by inviting the user to see the product and allowing the candidate to demonstrate the system, or by taking the system to the user involved. There should be an implementation plan written, including details of system changeover, training required and details of user testing.

1–2 Details of system changeover have been documented with some recognition that the user(s) will require training. Some evidence of user testing is given, usually by questionnaire or written comments by fellow students or others who were not directly involved in the development of the system.

3–4 A good implementation plan with details of training required. There is written evidence available from the third-party user indicating that they have seen the system in operation.

5–6 A clear and detailed implementation plan, including detailed stages of user testing. All aspects of user testing, user acceptance, implementation and system changeover have been documented. There is written evidence available from the user that they have used the system and agree with the strategy for implementation.

(iii) Appropriateness of structure and exploitation of available facilities [3 marks]

Some discussion of the suitability of methods and any product (e.g. hardware or software) used for the particular system should be included. Some recognition and discussion of the problems encountered and actions taken when appropriate should also be included. A log of such problems should be kept.

1 Some attempt at discussing the suitability of the hardware and software.

2 In addition, an informative diary, or log, of problems encountered has been kept.

3 A complete discussion of the hardware and software available and how they were suitable in solving the given problem, together with a good, informative explanation of the problems encountered and how they were overcome.

(d) Documentation [Total: 12 marks]

(i) Technical Documentation [6 marks]

Much of the documentation will have been produced as a by-product of design and development work and also as part of writing up the report to date. However, a technical guide is a **stand-alone** document produced to facilitate easy maintenance and upgrade of a system. The contents of the guide should, where relevant, include the following: record, file and data structures used; database modelling and organisation including relationships, screens, reports and menus; data dictionary; data flow (or navigation paths); annotated program listings; detailed flowcharts; details of the algorithms and formulae used. All parts of the guide should be fully annotated since this is important for subsequent development of the system. The specifications of the hardware and software on which the system can be implemented should be included.
Since the system in the technical guide will differ from one project to another, professional judgement as to what would be necessary for another analyst to maintain and develop the system has to be made.

1–2 Some items are present with some annotation attempted.
3–4 One or two omissions, but the rest is present and annotation is used sensibly.
5–6 No major omissions, with all parts fully annotated. For full marks the guide should be well presented rather than just a collection of items.

(ii) User [6 marks]

Clear guidance, as friendly as possible, should be given to the user for all operations that they would be required to perform. These would include input format with screens displays, print options, back-ups (file integrity routines), security of access to data and a guide to common errors that may occur. (Note the candidate would not be required to copy out large volumes of any underlying software's user guide, but to produce a non-technical and easy to follow guide for someone with little computer knowledge.) Some mention here of the relationship between items of software and the data they deal with may be relevant.

The user guide should be well presented with an index and, where necessary, a glossary of the terms used. Alternatively, an electronic guide could be based around hypertext links (screen dumps will be required).

1–2 An incomplete guide, perhaps with no screen displays. Some options briefly described but difficult for the user to follow.
3–4 All but one or two options fully described (for example, back-up routines not mentioned). In the main the options are easy for the user to follow with screen displays.
5–6 A full user guide with all options described well presented (possibly as booklet) with an index and a glossary. No omission of any of the options available (including back-up routines, guide to common errors). Marks may be lost for inadequate descriptions of some options. For full marks, good on-screen help should exist where this is a sensible option.

(e) Evaluation [Total: 8 marks]

(i) Discussion of the degree of success in meeting the original objectives. [3 marks]

This discussion should demonstrate the candidate's ability to evaluate the effectiveness of the completed system. The original objectives stated in requirements specification should be matched to the achievements, taking into account the limitations. User evaluation is also essential and should arise from a questionnaire or, preferably, direct user evaluation. For full marks it is important that the user provides sets of data as they are likely to occur in practice, and that the results arising from such data be given. This data is typical data rather than test data and it may show up faults or problems that the candidate's own test data failed to find.

1 Some discussion about the success, or otherwise, of the work, but with no reference to the specification set out in **(a) (ii)**.

2 Some discussion about a number of the objectives set out in **(a) (ii)**, but some omissions or inadequate explanation of success or failure.

3 A full discussion, taking each objective mentioned in **(a) (ii)** and explaining the degree of success in meeting them, indicating where in the project evidence can be found to support this or giving reasons why they were not met.

(ii) Evaluate the user's response to the system **[3 marks]**

It is important that the user is not assumed to be an expert in computer jargon, so some effort must be made to ensure that the system is user-friendly. It will be assumed that the user will have considerable knowledge of the underlying theory of the business being computerised. Clarity of menus, clear on-screen help and easy methods of inputting data are all examples of how the system can be made user-friendly. Here marks are awarded for the degree of satisfaction that the user indicates in the acceptance procedure. Could the system or its results be used? Was the system specification achieved? Do any system faults still exist? The candidate should evaluate the user's response to the final version of the system.
It is important that the user becomes an active participant in this section and that user responses are reported.

1 Some effort has been made to make the system user-friendly, but the user still has difficulty using the system.

2 The system is, in the main, user-friendly, but there is room for improvement (e.g. no on-screen help has been provided). The user indicates that the system could be used but there are some faults, which need to be rectified.

3 A fully user-friendly system has been produced. The user indicates that the system fully meets the specification given in section **(a)**, and there are no known faults in the system.

(iii) Desirable extensions **[2 marks]**

As a result of completing the system, the candidate should identify the good and bad points of the final system highlighting any limitations and necessary extensions to the system, indicating how the extensions could be carried out.

The candidate identifies clearly good and bad points and any limitations.
The candidate clearly portrays the good and bad points of the system indicating the limitations, possible extensions and how to carry out the extensions.

COMPUTING

Advanced GCE
Unit 9691/04: Computing Project

Candidate Record Card

Please read the instructions printed overleaf before completing this form. One of these cover sheets, suitably completed, should be attached to the assessed work of **each** candidate in the moderation sample.

Examination session	**June/November***	*please delete as necessary	**Year**	**2**	**0**	**1**	

Centre name	

Centre number					

Candidate name		**Candidate number**				

Assessment Criterion	**Mark**
Definition, Investigation and Analysis (max 11)	
Design (max 11)	
Software Development, Testing and Implementation (max 18)	
Documentation (max 12)	
Evaluation (max 8)	
Total (max 60)	

Authentication by the teacher

I declare that, to the best of my knowledge, the work submitted is that of the candidate concerned. I have attached details of any assistance given beyond that which is acceptable under the scheme of assessment.

Signature _____ Date _____

UNIVERSITY *of* CAMBRIDGE
International Examinations

INSTRUCTIONS FOR COMPLETION OF THIS FORM

1 One form should be used for each candidate.

2 Please ensure that the appropriate boxes at the top of the form are completed.

3 Enter the mark awarded for each Assessment Criterion in the appropriate box.

4 Add together the marks for the Assessment Criteria to give a total out of 60. Enter this total in the relevant box.

5 Sign and date the form.

COMPUTING 9691/04
Coursework Assessment Summary Form
A/AS Level

Centre Number						Centre Name		

Candidate Number	Candidate Name	Teaching Group/Set	Total Mark (max 60)	Internally Moderated Mark (max 60)

Name of teacher completing this form		Signature		Date						
Name of Internal moderator										